Full Name: Douglas Ryu Takada

WSU ID: 011766707

Eval:

For this function, I called parse line and saved the value to determine if when creating a new child if it the job should wait in the foreground or the background. From there I check if there are built-in commands passed in. If there are execute them immediately in the builtin_cmd function. Otherwise, it is executable. Depending on the ampersand at the end, create the child and make it a foreground or a background process. For creating the jobs, I called the add jobs and waitfg functions to do the job management.

Builtin_cmd:

This function, if the command name matches quit, sends a sigquit signal. If the command name matches jobs, call the listjob helper function to print out the job list. Finally, if the command name matches either fg or bg, call the do-bgfg function to implement the process changes.

Do_bgfg:

For this function I created a temporary job struct to store the desired job to change its state and to send a signal to its respective process ID. The argv[1] grabs the second argument that should be the job ID. From here I do error checking to make sure that it is not empty, and an integer. I also do pointer arithmetic so that I only grab the number instead of the percentage sign in front of the number along with it. If it is not in the correct input, I will print out the respective error statements as shown in the code. If it is the proper input, then I will update the respective job state, and send the signal to begin the process again. If it is a fg, then I will call the waitfg function, so the program waits until the job is complete. Otherwise exit the function.

Waitfg:

This function takes in the process id of the respective job, check to see if it is the same as the foreground process ID. If it is, then calling the waitpid function to check if the job has either been moved to a background process or if the job has completed. If either of these is the case, then delete the job with that process id and exit. If these conditions have not been met, then stay in the while loop and keep checking if the current process id is the same as the foreground process id.

Sigchld_handler:

This function allows for the reaping of all the zombie children. The –1 in the waitpid means that it will wait for any child process to complete and then send a SIGTSTP signal to reap the child. The second

argument I put as 0 since the signal is not important here. The WNOHANG option means that if there are no zombie children, return immediately.

Sigint_handler:

For this function, I called the fgpid to grab the process id number of the foreground process so I could send the process the SIGINT signal through the kill command. I used the - in front of the tmp to send the singal to the entire group with the same process id. Then I printed that the process was stopped.

Sigtstp_handler:

Finally, for this function, I grabbed the job that is the foreground job using the getjobpid and the fgpid helper functions. From here I update the state of the job, send a SIGTSTP signal through the kill function, and lastly print out the result of what happened.